Network Failure Analysis for CENIC

Abstract—

***Abstract*—The network services today are having an significant increase of the requirement on the availability of the network, while at the same time the growing size and comple**

I. Introduction

As more and more enterprises heavily rely on the networks, there is clearly an increase in importance to provide better availability or even uninterrupted service more aggressively. However, it brings us a much challenging work to deliver such promise as we are building much more complex and larger networks comparing to years ago. Although tons of research projects have been trying to work it out, it is still not very well solved. As well as many previous work shown [2]–[4], we believe the better analysis of the network failures should be considerably helpful for us to understand how the failures occur, how they affect the network behaviors and how to eliminate them in order to provide better availability.

To deliver such network failure analysis in practice, one will need much information about the causes, time lasted, influence to the network and many other specifics that are clearly not intuitively provided by the network protocols in use currently. Thus, the traditional approaches [1], [5] for doing this kind of analysis is to build some extra software or hardware support, which would incur large amount of expense and perhaps performance overhead. Consequently, most of such failure analysis are performed in the research community. To our knowledge, California Fault Lines [6] is the first piece of work that trying to extract these information from commonly used production networks today.

Other than reconstructing historical network failure events, we are able to perform further characterization and analysis on those failures with very similar "cheap and dirty" data which we can easily obtain from today's networks. To better understand the network behavior, we describe and validate a methodology to map the routing changes to corresponding ISIS failures. In addition, we propose an approach to better understand how the failures occur and how they affect the network behaviors. Furthermore, characterization and discussion are presented about the unexpected network behaviors including routing loops and non-existent links.

Specifically, we set up six source machines respectively at UC Berkeley, UC San Diego, UC Los Angeles, UC Santa Barbara, UC Davis and UC Santa Cruz, all in CENIC (the Corporation for Education Network Initiatives in California) which is the autonomous system we are trying to analyze. Each of the six machines are supposed to send a series of traceroute requests to the end hosts of the failing link when it is detected with the help of syslog messages automatically sent by CENIC. Firstly, we map the traceroute data back to corresponding network failures for further study, and validate how well the failure detection and traceroute are working based on the mapping. Secondly, we detect the routing loops and non-existent links in the traceroute data, and carry out some statistics and characterization on these unexpected network behaviors as well as the ISIS failures. At last, with the help of link weights of CENIC, we evaluate how well the network protocols work on routing and analyze how the failures affect the network behaviors.

The rest of the paper is organized as follows. We begin with the discussion of related work in Section II. Then the data we use in this work is introduced in Section III. Section IV presents the our methodology for the characterization and analysis, and they are validated in Section V. We present our analysis in Section VI and conclude in Section VII.

II. Related work

Network failure analysis has long been an interesting topic since the born of the Internet [**?**]. Researchers have done a lot work in charaterizing all kinds of network failure [**?**], [**?**], [**?**], [**?**], [6]. They see network failures from different perstives.

In [**?**] Athina and his colleagues analyzed failures in a IP backbone operated by Sprint. Their analysis are based on the IS-IS data they get from the backbone network. They categorized these failure data by cause, and obtained some statics on these data. However, they did not dig into the acutal character of each failure.

While in [**?**], [**?**], researchers carried detailed analysis under large enterpise data centers. In [**?**], Shaikh and his colleagues focused on OSPF behavior. They tracked 205 router for a month, described how LSAs works, and gave corresponding explainations. Although they are intended to gave a clear illustration of how LSAs behaves, they offered a valuable insight of network failures at the same time. On the other hand, in [**?**], Phillipa and his colleagues aimed to find out how failures behaves inside a larege enterpise data center. Since they get to access the data center directly, they are able to classify all kinds of hardware failures and software failures. They gave a lot of statics about how failures happens, and how redundancy performs in dealing with these failures.

The work mentioned above all involves accessing large scale of network infrastructure. In [6], Daniel together with other researchers did a detailed analysis over the CENIE network in california, with only the syslog history, mailing list and configuration setting of the routers. They successfully

established correlations between these datas and found some interesting statics of the network.

In this paper, we continue to work on the data extracted from CENIC network, on a more detailed granuality, and more focus on the automated data instead of messages like mailing lists.

III. Data sources

We delivered this piece of work in CENIC, from which we got several sets of data. In order to set the context for our further analysis and characterization, we described the particular data sources we used as follows.

A. The CENIC Network

The Corporation for Education Network Initiatives in California (CENIC), was a state-wide network providing Internet access to California's education and research communities. [**?**] It had a combined enrollment of more than six million members including the University of California system, California State University system, community colleges, and K-12 school districts.

B. ISIS Failure

This failure data was detected by the underlying routing protocol based on the adjacency status. The protocols running across the ISs required the routers to send and receive *hello* messages. By default, each router were supposed to send a *hello* message to its neighbors every 10 seconds. And if one router hadn't heard from one of its neighbors for 30 seconds, it was declared as disconnected.

For each of these failures, it followed the format shown below. The fields $failure_{start}$ and $failure_{end}$ were formatted in UNIX timestamp.

$\{router_1, port_1, router_2, port_2, failure_{start}, failure_{end}\}$

C. IP-Router Mapping

As different IP addresses were assigned different routers at different time, there had to be a table describe the mapping between IP address and router at some point of time. And each record contained a field showing the IP address assignment was valid until a certain time.

For each record, it followed the format shown below. The field $time_{valid\_till}$ was formatted in UNIX timestamp.

$\{IPaddress, router, port, time_{valid\_till}\}$

D. Link Map

The link map described the available links in CENIC and their valid time period. Each of the record contained the two routers and two ports the link connected, and the valid time period.

For each link, it followed the format shown below. The fields $link_{start}$ and $link_{end}$ were formatted in UNIX timestamp where the links that are still valid right now were marked huge numbers in $link_{end}$.

$\{router_1, port_1, router_2, port_2, link_{start}, link_{end}\}$

E. Link Weights

The link weights were presented with the two routers it connected and one integer weight. The weights were typically assigned to the links by configuration files, and the values were not consecutive.

For each record, it followed the format shown below.

$\{router_1, router_1, weight\}$

F. Traceroute

We had 6 machines set up at different locations in CENIC, and they were supposed to send traceroute request to the ends of the failure link when it was detected. The reason why we had 6 of them was because the more source machines we had, the more likely that we could get aware of the routing changes incurred by the link failures, because different sources would send the requests from different directions of the network and some of them might not route through the failure link so they wouldn't be affected by it actually. Those machines were located respectively in UC Berkeley, UC San Diego, UC Los Angeles, UC Santa Barbara, UC Davis and UC Santa Cruz.

The traceroute data were simply the original output of the traceroute application. However, there were two issues we needed to pay more attention in order to get more accurate characterizations.

1) Hops with local area network Typically, the first few hops of the traceroute were within the local area network, hich we were not actually interested in. Consequently, we could verify whtether the hop was within LAN or in CENIC by looking up if there was an IP address assigned to it.
2) Unrecognizable hops Not all the routers in CENIC support the UDP traceroute application, so some of them would just reponse as ** when they were traced. We were not able to get rid of it, thus we had to be careful with it in the following analysis.

IV. Methodology

A. Reconstruction of the Network History

Before doing any other analysis or characterizations, we needed to reconstruct the network history and topology from the data sets we had. First of all, we parsed the raw data of ISIS failures, mapping of router and IP address and available links within CENIC. Most of these data were timely, that for instance one IP address could be assigned to many other routers or end hosts after it expired for one router. So we stored these data with timestamp at the granularity of one second, which was exactly the granularity provided by these data, and kept track of them timely.

Then we parsed the traceroute data, which was a little bit more complex since there were many trivial issues we needed to take care. For most of the traceroutes, the first few hops were not actually the routers or end hosts in CENIC, instead they were basically the routers within the local area network of which we would like to get rid. In addition, there were some routers in CENIC that didn't support UDP traceroute,

which meant the traceroute could give us nothing more than ** since the router was not recognizable. And there was also an issue about the destination of traceroute, that it turned out that the traceroute stopped when the reach the same router of the destination regardless of the port indeed. So we would need to check whethre the last hop of traceroute and the destination are on the same router rather than exact same port and IP address.

B. Mapping between ISIS Failures and Traceroutes

We would like to map the ISIS failures to corresponding traceroutes and vice versa not only to validate how the failure detection and traceroute were working, but also for further characterization and analysis. In order to do so, we had two rules to establish the correlation between them.

1) Time Consistency The occurrence of the failures and their corresponding traceroutes should be fairly close in time. In particular, because of the possible 30-second delay of our failure detection mechanism, we had to widen the mapping window on time consistency.
2) Route Consistency This one was a little bit trickier because the destination of the traceroute and the failure link could not be the same port and IP address, so we had to check whether they were supposed to be on the same router.

C. Non-existent Link Detection

We then extracted the nonexistent links from the traceroute data, as a part of the unexpected router behavior. We traversed through all the traceroute data, assuming there should be a link between any adjacent hops in the record. This is verifried by looking up in the linkmap to see if there is a link between these two routers at that time. Although it seemed to be fairly straight forward, we have to apply some constraints to the process in order to avoid useless data.

1) Look up by router We say there is a link between the two hops if there is a link betwee the two routers on the hop, without considering the port. Since the weights used to calculate path in the network is given by router, ignoring ports would eliminate useless data. In addition, traceroute check router, instead of router with port, to see if it has reached the destination too.
2) No respones During this process, we ignored those hops that failed to respond to traceroute, which appeared in the record in the form of "* *", as well as those routers outside CENIC network.

D. Routing Loop Detection

We also extracted loops from the traceroute data, as the other part of the unexpected routing behavior. We parsed all the traceroute records and detected loops following two different rules.

1) Verify by IP We could verify whether a traceroute record contained loop by checking if there existed an IP address appearing more than once.
2) Verify by router We could verify whether a traceroute record contained loop by checking if there existed a router appearing more than once. This approach should at least return all records that IP-approach returned.

E. Routing Performance Characterization

Basically, we meant the performance on calculating the shortest routing path by saying routing performance. Given the link weights of each links in the CENIC, we were able to calculate the sum of the weights for each of the traceroute record while regardless of the first few hops within the local area network, as well as the weight of the global optimal one. However, there were some issues we needed to pay attention for both of these calculations.

1) Actual route There might be some of the middle hops were not recognizable because lack of the support of UDP traceroute. For these records, the best effort we could do is to calculate the weight of the sub-route. However, this didn't make much sense in practice because there was a great probability that the actual route and optimal one had totally different sub-routes, so that they were not comparable at all.
2) Optimal route Here we just simply implemented the Dijkstra's Algorithm to calculate the optimal shortest path.

In addition, we had to get rid of the first few hops within local area networks for both optimal route and actual one since we didn't have the link weights of them.

V. Validation

A. Network History

Assuming the active probing of the certain network could give us a complete history of it, we had this history of the CENIC from 10/20/2012 to 02/09/2013. There were 40074 ISIS failures during this period in total.

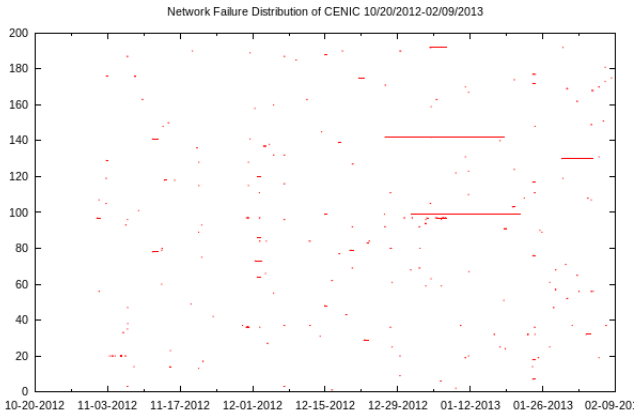

Fig. 1. Network failures distribution of CENIC from 10/20/2012 to 02/09/2013, where Y-axis is the failure link. Most of the failures lasted for a fairly short time period and there were 192 links had failed at least once during this time period.

Since we would like to do a comparison among the route before, during and after the failure to better understand how

failure affects the network behaviors, it was useful to find out how long each failure last for. Thus, the CDF of failure recovery time was shown in Figure 2. Surprisingly, there were 77.87% of the failures that were recovered in 0 seconds, which means they didn't actually affect the network in the granularity of second that we used. And more than 95% failures were recovered within 6 seconds, which could potentially make the corresponding traceroutes much less interesting.

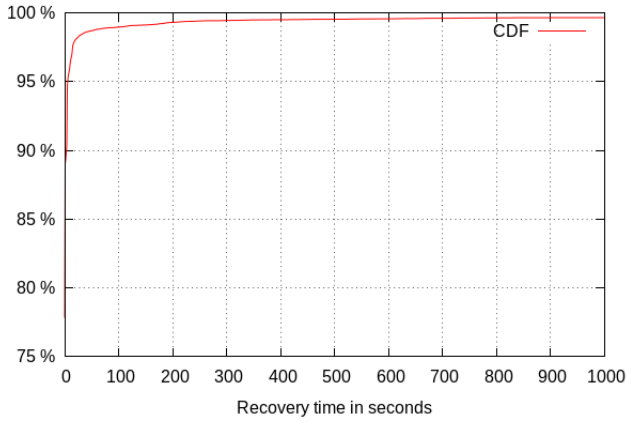

Fig. 2. Network failure recovery time CDF of CENIC where more than 95% failures recovered within 6 seconds.

Because of the relatively short recovery time, it is very likely that the failure had already recovered before we sent very few traceroutes. Thus it could reduce the amount of data we were interested in because most of the traceroute would not even aware of the route changes due to the large granularity. However, even with small percentages, we didn't get very tiny numbers in the following analysis because of the fairly large number of failures we had in total.

B. Network History Reconstruction

To validate the network history we reconstructed from the traceroute data, we tried to map the probed failures to corresponding traceroutes. Based on the link and router map of CENIC we had successfully mapped 40073 ISIS failures to corresponding traceroute data out of the total 40074. The only one that had not been mapped was because we didn't have the IP address of the link. Thus, we believed it is save to say our failure detection works pretty well from this point.

For these 40073 detected failures, we had pinged at least one end of the failure link. And for 39505 out of 40073 (98.58%) failures, we had pinged both ends at least once. Consequently, we believed our data is very representative for our analysis.

C. Statistices of Unexpected Network Behavior (Loops & Non-existent Links)

80 nonexistent links out of 70831 traceroute records. They are classified into 5 categories, 45 in normal route shift, 9 in complex route shift, 3 in shortcut, 15 in trapped in loop, 8 in cannot verify. Among all these 80 nonexistent links, only 8 are directly involved with failures, and 18 of them happens when failures occured in the network other then themself.

D. Measurement Bias

First of all, the granularity of the network failures and the traceroute data could potentially affect our analysis of the network behaviors. As discussed ealier, the ISIS failures were probed at the granularity of one second, while many of them lasted less than couple seconds actually. This reduced the number of interesting traceroutes since they could not even reflect the effects of such short failures. We believed that a much finer granularity of the failures would be very helpful for further analysis.

Additionally, our failure detection mechanism is based on adjacency status reported by the underlying routing protocol. For example, to ensure connectivity, the IS-IS protocol requires routers to send and receive hello messages. By default, a router sends a hello message once every ten seconds and declares a link disconnected if no hello message is received for thirty seconds. Consequently, the failure detection could have up to 30 seconds of delay, which could reduce the accuracy of the mapping between the failure and corresponding traceroutes. In addition, this would probably affect our analysis about how the failures affected the network behaviors.

VI. Data Analysis

A. Loop Characterization

In the 70831 traceroute records, we detected 366 records by IP approach and 395 records by router approach that contain loops in total. yesi For IP approach, we categorized these 366 records by the length of loop and found that more than 95% (350 out of 366) of loops were jumping between only 2 routers. The longest loop had a length of As of the long loop records, at least 11 of 16 would also fall into a two-router trap finally. We also found that there were onl 11 rcordes fthat naally arrived at the destination. aAll of the 11 records shared loop length of 2. 9 of 11 records only experienced 1 round of loop. Another interesting phenomenon was that 344 of 366 records were trapped into a loop that consisted of 2 adjacent IP addresses.

For Router approach, we categorized these 395 records by the length of loop and found that more than 90% (359 out of 395) of loops are jumping between only 2 routers. As of the long loop records, at least 24 of 36 would also fall into a two-router trap finally. We also found that there were 29 records that finally arrived at the destination. 26 of the 29 records shared loop length of 2 while 2 records had length of 4 and the last one 6. 26 of 29 records only experienced 1 round of loop.

We then tried to figure out a mapped from loop records onto IS-IS failures to reveal what caused such undesired routing behavior. We categorized the loop records by the potential pattern of failures' impact. By applying both time consistency and router consistency, we obtained the first category, Simple Link failure. However this pattern could only explain a small part of the loop records. Introducing a failure propagation assumption by loosing the time constraints granted us the second category, Complex Failure Propagation. Potential causes for the rest part

of records that couldn't be explained by the above two reasons might be done in the futurte work. For the following part, we only consider the 366 records we obtained by IP approach.

Simple Link Failure

Suppose a trace-route arrived at destination without any unexpected failures. A simple link failure may occur in any link on the normal path. Then the router would have to forward packets to other routers which actually could not help reach the destination and finally lead to a loop.

Complex Failure Propagation

While considering the direct influence of failure in the above part, it was also possible that some failures might leave impact on the network even after they were revocered. Suppose a trace-route arrived at destination without any unexpected failures and then a failure occur in one link on the path. Then the routers ahead the failure node might detect that forwarding packets to the failure node was useless. Finally these routers would try other ways. If unfortunately these ways didn't work either, then loop would happen again.

B. Non-Existent Links Charaterization

Among the 70831 trace-route records, there were 80 records has non-existent links in them. We classify these records into 5 categories, with respect to the complexity of their cause. They are: normal route shift, complex route shift, shortcut, trapped in loop, cannot verify. They are distributed as in figure:3

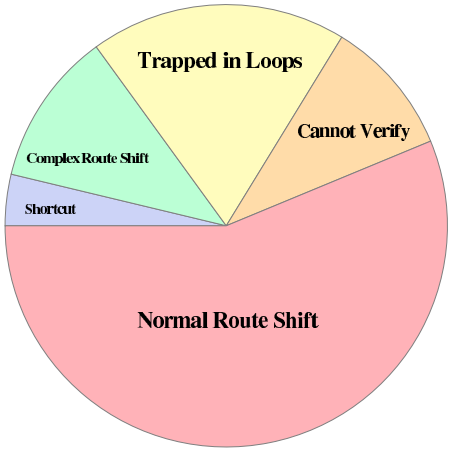

Fig. 3. Piechart of different types of missed links

Normal Route Shift

Consider a trace-route with max hops 5. In most situation, a certain route won't change during the process of trace-route. However, it is possible for router to choose another path during this process. An example is shown in figure: 4.

It is important to notice in figure:4 that, both of the two routes reached destination E, with C on the third hop and M on the fourth hop. We identify this kind of links by finding valid route with the same node on the same hop. 45 out of the 80 records fall into this category.

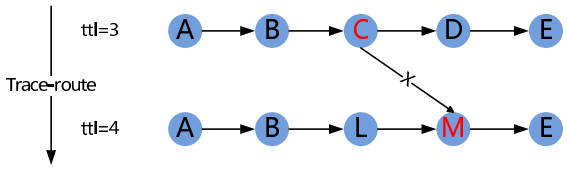

Fig. 4. A Normal Route Shift example

Complex Route Shift

This was a more complicated situation than in the first case. There were three key differences with the first category. First, a complex route shift may shift among more than 2 routes in a trace-route. Second, the picked route is allowed to be a route that did not reach the destination. Third, it is possible that none of other trace-route record can match a node on a certain hop of a complex route shift. Still, consider a trace-route with max hops 5. An example is shown in figure: 5.

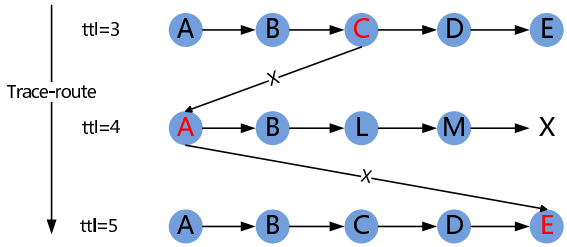

Fig. 5. A Complex Route Shift example

From figure:5, we cannot tell what exactly happened. Although we see node A on hop 4, we can see no A on hop 4 from any other record. This means, we have no idea of how and why trace-route reached B on the fourth hop. Hop 5 behaves similar with normal route shift. There were 9 records fall into this category.

Shortcut

Short cut is a special case of a normal route shift, with one of the route in the shift to be an invalid route. An example is shown in figure:6.

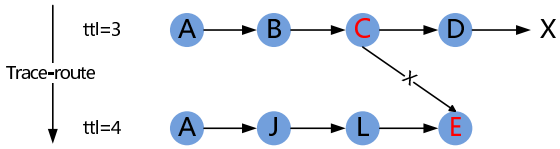

Fig. 6. A Shortcut example

To explain this phenomenon in a more intuitive way, suppose one of the router is down, trace-route cannot get to the destination due to the router failure. Then, during the trace-route, the router is fixed and back to work, trace-route got

the destination directly on the next hop. There were 3 records fall into this category.

Trapped in loop

This is another special case of normal route shift, with one of the route is a loopy route. An example is shown in figure:7.

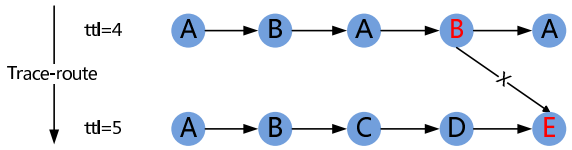

Fig. 7. Route Trapped in Loop example

There were 15 records fall into this category.

Cannot verify

The rest 8 records fall into this category. For these records, there exist some node that is never seen in other trace-route records with the same source and destination. Thus, there is no way we can tell what happened with such records.

In order to further analyze why these nonexistent links occurred, we tried to find out the relationship of nonexistent links with network failures.

26 out of the 80 nonexistent links happened at the same time as a failure some where in the network. But only 8 among these 26 records has one of it's nonexistent link involved in a failure. Thus, generally speaking, network failures were not the main cause of nonexistent links. Also, only 11 out of the 80 records with nonexistent links failed to reach the destination. In conclusion, nonexistent links occurred in a very small fraction of data, and has no great impact on the performance of network, since most of them arrived at the destination anyway.

C. Failure Analysis

Given the traceroute data, we were able to take a closer look at the ISIS failures, which could potentially help us better understand the network behaviors and then try to reduce the number of failures. From all the 40073 failures we've ever pinged, only 125 (3.11%) of them had ever caused a route change from at least one source. This was a relatively small number comparing with the total failures we detected, which I thought is mainly because we only had 6 sources to send traceroute requests and there are only 3 of them worked as we expected, as well as most of the failures lasted for very short time period. It would be considerably helpful if we could have more sources sending traceroute to the failure end hosts, which would probably give us more interesting routing changes.

And about half of these changes, 63 out of 125 (50.40%), were not detectable because we could only get more or less unrecognized hops displayed as ** from traceroute. Since we couldn't recognize them, we didn't have their route weights and were not able to draw any conclusions on how their routes change quantitatively. But we could still do some analysis about the accessibility of the failure ends.

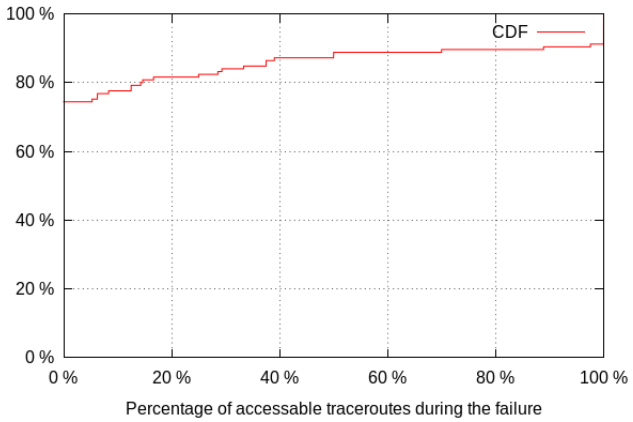

Fig. 8. Percentage CDF of accessible pings during the link failure time. 74.40% of the 125 failures which had caused route changes hadn't affected the accessibility of the end hosts, which meant the routing protocaol worked pretty well during those failures.

It was clear in Figure 8, that the end hosts of 93 out of 125 (74.40%) failures have kept unreachable during the whole failure period. Because all these 125 failures had caused route changes, which is to say that all these routes were affected by the failures somehow, we had to say that the routing protocols were performing relatively poor. However, there were still 8.80% of the failures had never affected the availability of the ends of the failure links during the whole failure period.

Furthermore, we did some statistics for the failure links, where a few links (192) occurred large number of failures (40073). From Figure 9, we could easily find that most of the failure links had small numbers of failures ever occurred. But the most surprising part of this figure is that there are some links that had more than thousands of failures during this period of time, which were supposed to be taken care of after tens of frequent occurrences.

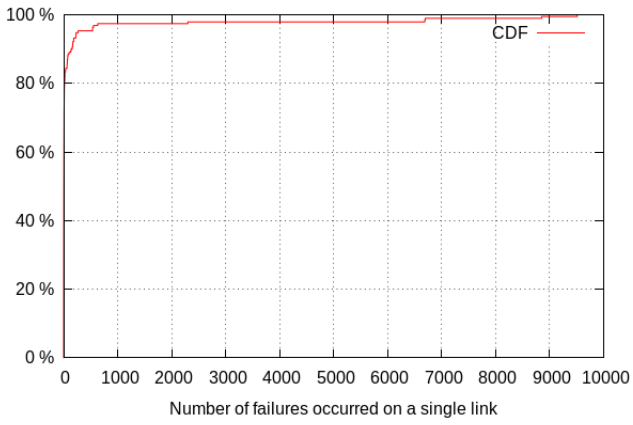

Fig. 9. The distribution CDF of the failure links. Most of the failure links had very small numbers of failures, where 74.48% of them had occurred less or equal 10 failures. However, there are some links had very frequent failures that were more than 8000 times during this period.

D. Routing Performance Characterization

With the help of the link weights, we also did a characterization on how well the routing protocols were doing on calculating the shortest routing paths. Among the 70831 traceroute records in total, we successfully extracted 52431 (74.02%) recognizable traceroutes. The rest of the traceroutes were unrecognizable because those router in CENIC did not support UDP traceroute, which meant we could only get ** as transmitting through these routers.

Fig. 10. The actual route weight compared with the optimal route weight calculating with the link weights. 48426 out of 52431 (92.36%) traceroutes successfully chose the global optimal route, which meant the routing protocols were working relatively well on this.

In Figure 10, it ware clear that most of the actual path weights were already the optimal ones, while there remained 7.64% were not. We would like to further analyze how bad those local optimums were, but it didn't make sense to compare the actual weights since the link weights were not consecutive at all.

We had also tried to carry out analysis about how the failures affected the network routes and their performance. In order to do that, we would need to understand the pattern manually before we could tell the program how to do so. The biggest issue we encountered was to tell why the routing protocols responded like this and the correlation between routing changes and the failure links from thousands of records. Large amount of manual recognition and classification was inevitable while we were so limited on time, so we had to leave this to the future work.

VII. Conclusion

In this paper, we show an approach to extract the network failure information of CENIC with fairly cheap and commonly applicable mechanism without expensive probing infrastructures. Then we propose a methodology to analyze and characterize those extracted failures and detect the unexpected network behaviors including routing loops and non-existent links. And we also carry out a study on how the routing protocols are working on calculating the shortest paths. We believe our mechanism is fairly general and applicable for most of the common production networks.

Acknowledgment

We'd like to thank Prefessor Snoeren and Siva's advice on this project. And we do really appreciate the help from Dr Daniel Turner who provided most of the data and patiently answered every single question we asked. Unfortunately, this project is not funded by any fundation.

References

[1] Ítalo Cunha, Renata Teixeira, Darryl Veitch, and Christophe Diot. Predicting and tracking internet path changes. *SIGCOMM-Computer Communication Review*, 41(4):122, 2011.

[2] Craig Labovitz, Abha Ahuja, and Farnam Jahanian. Experimental study of internet stability and backbone failures. In *Fault-Tolerant Computing, 1999. Digest of Papers. Twenty-Ninth Annual International Symposium on*, pages 278–285. IEEE, 1999.

[3] Athina Markopoulou, Gianluca Iannaccone, Supratik Bhattacharyya, Chen-Nee Chuah, Yashar Ganjali, and Christophe Diot. Characterization of failures in an operational ip backbone network. *IEEE/ACM Transactions on Networking (TON)*, 16(4):749–762, 2008.

[4] Venkata N Padmanabhan, Sriram Ramabhadran, Sharad Agarwal, and Jitendra Padhye. A study of end-to-end web access failures. In *Proceedings of the 2006 ACM CoNEXT conference*, page 15. ACM, 2006.

[5] Vern Paxson. End-to-end routing behavior in the internet. *Networking, IEEE/ACM Transactions on*, 5(5):601–615, 1997.

[6] Daniel Turner, Kirill Levchenko, Alex C Snoeren, and Stefan Savage. California fault lines: understanding the causes and impact of network failures. *ACM SIGCOMM Computer Communication Review*, 40(4):315–326, 2010.